SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1596

SUSTAVI ČESTICA U SIMULACIJAMA MNOŠTVA I JATA

Marin Maršić

Zagreb, lipanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 9. ožujka 2018.

DIPLOMSKI ZADATAK br. 1596

Pristupnik: **Marin Maršić (0036477535)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Sustavi čestica u simulacijama mnoštva i jata**

Opis zadatka:

Proučiti načine simuliranja ponašanja mnoštva i jata objekata te mogućnosti upravljanja njihovim ponašanjem i postizanjem željenih oblika formacija objekata. Proučiti mogućnosti upotrebe sustava čestica u navedenim simulacijama. Razraditi simulacije koje će prikazivati ponašanje jedinki u postizanju željenih ciljnih formacija. Na raznim primjerima prikazati ostvarene rezultate. Diskutirati utjecaj različitih parametara. Načiniti ocjenu rezultata i implementiranih algoritama.
Izraditi odgovarajući programski proizvod. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 29. lipnja 2018.

Mentor:
Prof. dr. sc. Željka Mihajlović

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:
Prof. dr. sc. Siniša Srbljić

Sadržaj

1. Uvod 1
2. Algoritam kretanja jata 3
   2.1. Sustavi čestica 3
   2.2. Boidi 3
      2.2.1. Pravilo kohezije 4
      2.2.2. Pravilo separacije 4
      2.2.3. Pravilo poravnavanja 5
      2.2.4. Praćenje cilja 5
      2.2.5. Praćenje putanje 5
   2.3. Optimizacija algoritma 6
      2.3.1. K-D stabla 6
      2.3.2. Nepotpuno ažuriranje 8
3. Animacija kretanja jata 9
   3.1. Rojenje jata 9
   3.2. Praćenje putanje 9
   3.3. Objektom definirana formacija jata 11
   3.4. Transformacije oblika 13
4. Implementacija i rezultati 15
   4.1. Učitavanje objekata i definiranje putanje 15
   4.2. Definiranje boida 15
   4.3. Ažuriranje položaja čestice 17
   4.4. Predator 17
   4.5. Performanse 18

5. Zaključak 20

6. Literatura 21

1. Uvod

Jata ptica u zraku, riba u vodi, rojevi pčela ili krda divljih životinja ostvaruju vrlo složena skupna kretanja koja često privlače pozornost ljudskih promatrača. Jato se ptica kreće nebom kao da je upravljano nekim središnjim dirigentom. Ptice zajedno mijenjaju smjer, a pri nailasku grabežljivca dolazi do usklađene nagle promijene smjera kretanja. Promatrajući takva kretanja, ornitolog Edmund Selous u 19. je stoljeću smatrao da su takva složena kretanja jata ptica posljedica zajedničkih misli koje ptice međusobno dijele.

Danas se, međutim, zna da pticama nitko ne upravlja. Svaka ptica donosi odluku o svome smjeru kretanja na temelju kretanja ptica u svojoj okolini pri čemu se smatra da jedna ptica promatra do oko sedam svojih susjeda [1].

Zbog svoje atraktivnosti, ovakva se kretanja često žele ostvariti u filmovima i video igricama. Korištenje pravih životinja često zna biti jako skupo, zahtjevno, a često je i nemoguće ostvariti neka složenija kretanja. Zbog navedenih razloga postoji potreba za računalnom simulacijom ovih kretanja.

Prvi koji je opisao algoritam za simulaciju jata ptica bio je Craig Reylnolds [2]. Svoj je algoritam temeljio na multiagentskom pristupu gdje svaki agent samostalno donosi odluku na temelju promatranja nekolicine agenata iz svoje bliže okoline. Njegov je algoritam pružao jednostavan način za ostvarivanje složenih kretanja te je postao temelj za sve buduće radove.

Na tom radu temelji se i ovaj. U ovom će radu biti opisana animacija sustava čestica unaprijed definiranom putanjom uz dodatak omeđenosti jata složenim oblicima. Cilj je postići ponašanje koje je što bliže onome u prirodi, a koje je presloženo da bi ga dobili prirodnim putem. To su ponašanja poput poprimanja složenih formacija, te transformacija u neki zadani oblik.

U prvom se poglavlju ovog rada opisuje osnovni algoritam za simulaciju kretanja jata, navode se izmjene uvedene u odnosu na osnovni algoritam te optimizacija algoritma. Specifičnosti vezane uz animaciju i njene faze opisane su u drugom poglavlju. Također, opisuju se i načini korištenja istog algoritma za postizanje različitih vrsta kretanja i

ostvarivanja formacija. U trećem poglavlju ovog rada opisuje se programska implementacija navedenih algoritama te se analiziraju njihova svojstva. Programsko rješenje implementirano je uporabom OpenGL biblioteke u programskom jeziku Java.

2. Algoritam kretanja jata

2.1. Sustavi čestica

Sustavi čestica tehnika su korištena u računalnoj animaciji kod koje se složeni efekti postižu mnoštvom malih jednostavnih grafičkih elemenata. Takvi se sustavi često koriste u ostvarivanju efekata kao što su vatra, dim, oblaci, zviježđa, galaksije i mnoštvo drugih. Čestice koje čine sustav djeluju temeljem parametara sustava te međudjelovanja s ostalim česticama. Čestice mogu imati definiran životni vijek, dob, veličinu, poziciju, smjer kretanja, brzinu te mnogo drugih parametara. Za ostvarivanje kretanja jata sustavom čestica potreban nam je konstantan broj čestica tijekom cijele faze simulacije. Sve čestice će imati jednak vijek trajanja koji odgovara trajanju simulacije, dok će ostali parametri dinamički ovisiti o trenutnom stanju u okolini svake čestice.

2.2. Boidi

Prvi opisani algoritam za simulaciju kretanja jata objavio je Reynolds u svom radu [2]. On je svoj algoritam temeljio upravo na sustavima čestica, a elemente tog sustava nazvao je boidima. Svaki boid je agent koji donosi odluke o svom kretanju na temelju svoje okoline vodeći se zakonima koji su unaprijed definirani za cijeli sustav, te parametrima svakog pojedinačnog boida.

U sustavu su definirana tri osnovna zakona koja upravljaju ponašanjem boida. To su zakoni kohezije, separacije i poravnavanja. Oni omogućuju postizanje kretanja koje nalikuje kretanju jata. Zakoni su opisani u nastavku. Dodatno su opisana i pravila praćenja cilja te praćenja unaprijed definirane putanje kojim se ostvaruje uvjetovanje smjera kretanja jata. Ta pravila omogućuju definiranje putanje kretanja jata koje u prirodi nije moguće ostvariti.

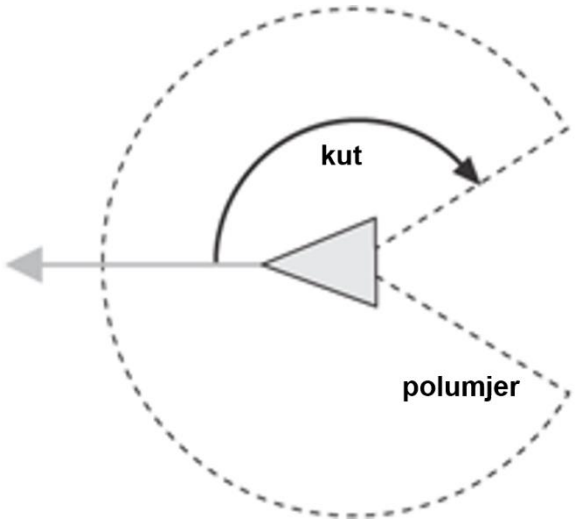
Susjedstvo pojedinog boida prostor je kružnog oblika unutar kojeg se nalaze njegovi susjedi. Svi boidi koji se nalaze unutar susjedstva nekog boida utječu na njegovo ponašanje, dok se utjecaj svih ostalih boida zanemaruje. Susjedstvo je definirano udaljenošću od središta boida te kutem koji određuje vidno polje i mjeri se od pravca kretanja boida. Prikaz susjedstva vidljiv je na slici 1.

2.2.1. Pravilo kohezije

Pravilo kohezije nastoji grupirati sve boide u jato. Boid se usmjerava prema prosječnoj poziciji svojih susjeda. Boid pod utjecajem ovog pravila teži pridruživanju skupini. Pravilo je skicirano za slici 2.1.

2.2.2. Pravilo separacije

Pravilo kohezije nastoji zbližiti boide, no potrebno je izbjeći da se oni previše zbliže. Kako bi spriječili da se boidi zabijaju jedni u druge zbog pravila kohezije, uvodi se protupravilo koje dovodi sustav u ravnotežu. Pravilo separacije nastoji razmaknuti boide i održavati ih na određenom razmaku jedne od drugih. Boid se usmjerava suprotno od smjera u kojem se nalazi prosječna pozicija njegovih susjeda. Zajedno s



Slika 2.1: Susjedstvo boida

prethodim pravilom, ovo pravilo omogućava da se boidi kreću u skupinama održavajući međusobni razmak. Ovo pravilo također je prikazano na slici 2.2.

2.2.3. Pravilo poravnavanja

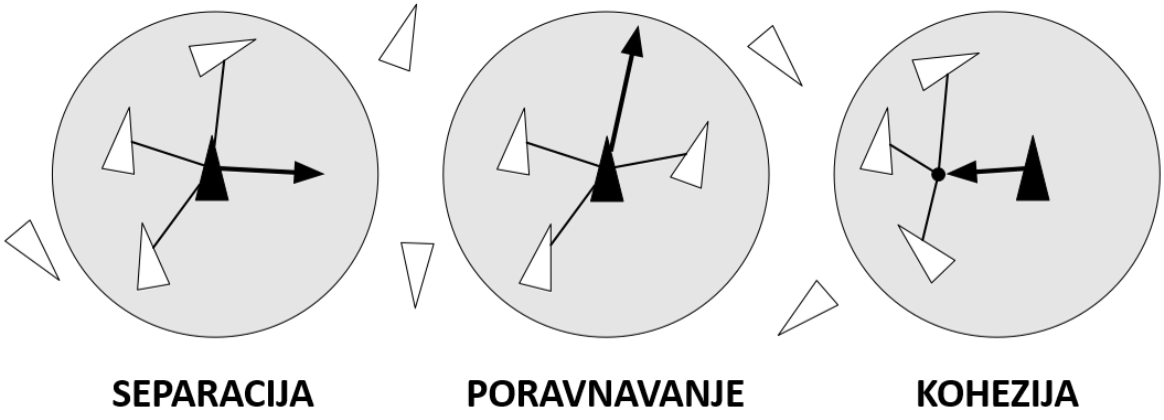
Pravilo poravnavanja nastoji ostvariti zajednički smjer kretanja svih boida. Smjer kretanja boida korigira se tako da više odgovara prosječnom smjeru kretanja njegovih susjeda. Ovo pravilo omogućava da se promjena smjera kretanja u dijelu boida propagira cijelim jatom te da se dva ili više jata pridruže jedno drugome kada naiđu jedno na drugo.

2.2.4. Praćenje cilja

Prilikom praćenja cilja boid korigira svoj smjer kretanja kako bi se približio cilju. Boid se usmjerava ka cilju dok god ga ne dostigne. Cilj se smatra dostignutim onda kada se boid nađe u njegovoj definiranoj okolini.

2.2.5. Praćenje putanje

Ako želimo ostvariti da se jato kreće zadanom putanjom, potrebno je odrediti način na



Slika 2.2: Pravila separacije, poravnavanja i kohezije

koji to želimo ostvariti. Reynolds je u radu [2] opisao nekoliko načina usmjeravanja boida poput usmjeravanja vektorskim poljem, usmjeravanje praćenjem vođe ili usmjeravanja praćenjem putanje. Ovo posljednje omogućava da svi boidi jata prođu definiranom istim putem. Pravilo je također implementacijski vrlo jednostavno za ostvariti te nije računski zahtjevno.

Praćenje putanje može se definirati kao praćenje niza ciljeva. Jednom kada boid ostvari svoj trenutni cilj, on se počinje usmjeravati prema sljedećem. U prostoru sustava definiraju se točke koje određuju putanju kretanja. Cilj je da svaki boid prođe kroz sve zadane točke istim redoslijedom. Kako nije poželjno da se oko ciljnih točaka stvaraju gužve, oko tih točaka definiraju se kružnice unutar kojih se boid treba nalaziti kako bi se smatralo da je posjetio tu točku. Povećavanjem radijusa kružnice krivulja putanje postaje blaža, dok se smanjivanjem radijusa kružnica povećava krivudavost putanje te zavoji postaju oštriji.

Kombiniranjem svih prethodnih pravila dobiva se jato koje se kreće definiranim putem. Ako se u kretanje jata želi dodati određena kaotičnost, to se može učiniti dodavanjem nasumičnog faktora svakom boidu koji ga usmjerava u proizvoljnom smjeru.

2.3. Optimizacija algoritma

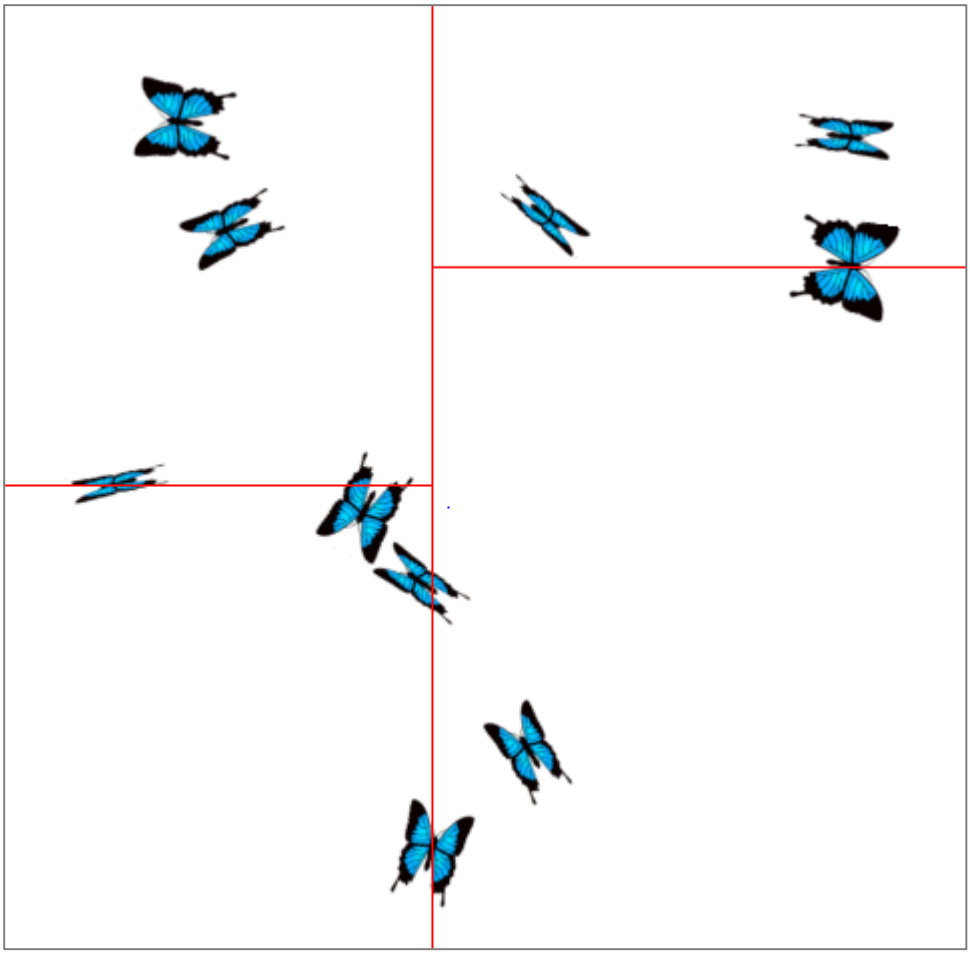
Sva prethodno opisana pravila određuju smjer kretanja boida na temelju njemu susjednih boida. Prilikom svake korekcije smjera kretanja boida potrebno je proći kroz sve boide u sustavu i pronaći njegove susjede te na temelju njih izračunati korekciju smjera. Ovakav pristup ima kvadratnu složenost te je nepraktičan za velike skupove čestica. Algoritam je moguće ubrzati na nekoliko načina.

2.3.1. K-D stabla

U slučaju da se boidi prostiru na velikom području, a susjedstvo je definirano malim prostorom, broj će susjeda nekog boida biti jako malen. Iz tog razloga bilo bi dobro podijeliti prostor na manje dijelove unutar kojih bi svaki boid tražio svoje susjede

umjesto da pretražuje cijeli prostor. Podatkovna struktura k-dimenzijskog stabla, ili skraćeno k-d stabla, omogućava upravo to. To je struktura koja rekurzivno svaki prostor dijeli na dva potprostora tako da se u svakome nalazi podjednak broj elemenata. Podjela prostora se naizmjence radi u svakoj od dimenzija prostora. Granica razdvajanja prostora prolazi pozicijom onog boida čija je pozicija medijan unutar tog područja. Struktura se gradi u svakoj iteraciji ponovnog izračunavanja pozicija boida. Iako izgradnja ove strukture zahtjeva određeno vrijeme, zahvaljujući njoj dolazi do velike uštede pri izračunima novih pozicija boida.

Na slici 2.3 možemo vidjeti dvodimenzionalno k-d stablo s dvije razine. Okomita crvena linija dijeli prostor na dva dijela s po pet leptirića sa svake strane. Svaki od ta dva potprostora dalje se dijeli vodoravnim linijama na prostore s po dva i tri leptirića. Pri svakoj podjeli prostora jedan se leptirić uvijek nalazi na granici dvaju prostora. Svaki od



Slika 2.3: Primjer dvorazinskog k-d stabla

leptira pri traženju svojih susjeda sada više ne mora provjeriti sve ostale leptire u sustavu, nego može provjeriti samo one koji se nalaze u njegovom potpodručju.

Ukoliko susjedstvo pojedinog boida definiramo kao njegovih n-najbližih boida, tada je izračun susjedstva moguće dodatno ubrzati algoritmom opisanim u radu [3]. Ondje se polazi od pretpostavke da u većini slučajeva, zbog zajedničkog smjera kretanja boida, susjedstvo iz jednog trenutka ostaje nepromijenjeno u sljedećem trenutku.

2.3.2. Nepotpuno ažuriranje

Polazeći od pretpostavke da većina boida iz jednog trenutka u drugi prelazi s minimalnom korekcijom smjera kretanja, moguće je ubrzati algoritam tako da se smanji učestalost ažuriranja smjera kretanja boida. Skup boida se podijeli u više podskupova. U svakom koraku nanovo se izračunava smjer kretanja samo za jedan podskup boida, a preostali dio boida zadržava smjer kretanja iz prethodnog koraka. Podskup boida čiji se smjer kretanja ažurira u svakom koraku ciklički se mijenja tako da svaki podskup bude ažuriran jednak broj puta.

3. Animacija kretanja jata

Kada imamo pravila koja omogućuju da simuliramo kretanje jata na način sličan onome u prirodi, sustavu možemo proizvoljno zadavati kretanje. Primjenom prethodno opisanih pravila i njihovom kombinacijom može se ostvariti mnoštvo zanimljivih efekata. U okviru ovog rada opisani su efekti rojenja, slijeđenja putanje, formiranje zadanog oblika te transformacija iz jedne formacije u drugu.

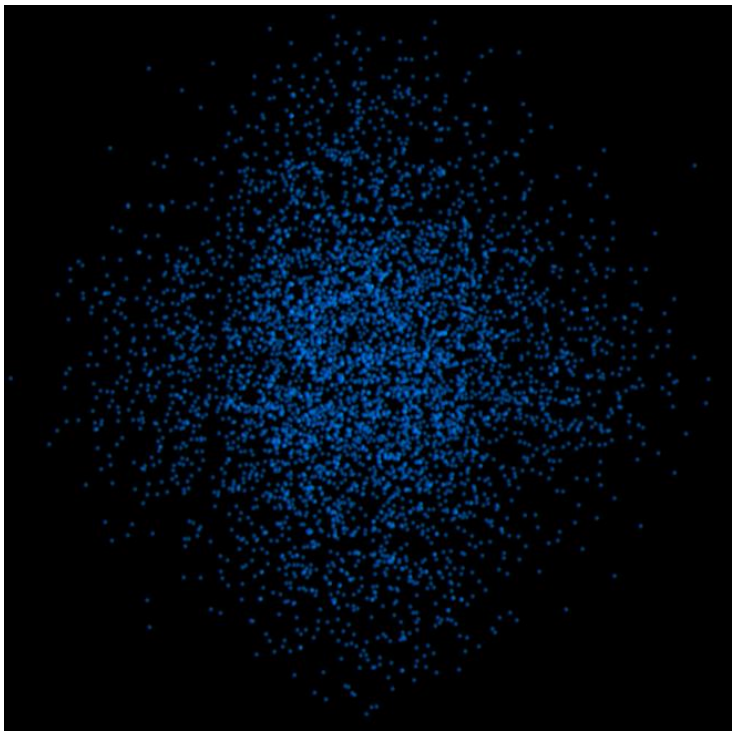
Svaki boid u sustavu definiran je svojom trenutnom pozicijom, smjerom kretanja i brzinom. Na svaki boid u svakom trenutku djeluju određene sile koje utječu na promjenu smjera i brzine kretanja. Te sile predstavljaju pravila kretanja boida opisana u prethodnom poglavlju. Zahvaljujući tome, nakon inicijalizacije sustava i definiranja pravila kretanja, nije potrebno dodatno utjecati na sustav, sve se unutar njega dalje odvija temeljem fizikalnih zakona.

3.1. Rojenje jata

Rojenje jata najjednostavnija je animacija. Sustavu čestica potrebno je definirati točku rojenja. To je točka koja privlači sve čestice sustava. Ona se ostvaruje pomoću jednostavnog pravila slijeđenja cilja. Ostala pravila za kretanje boida (kohezija, separacija i poravnavanje) omogućuju da se čestice kreću kaotično oko točke rojenja. Također, potrebno je ograničiti brzinu kretanja boida kako se on nikada ne bi zaustavio pri dostizanju ciljne točke, nego se nastavio kretati u njenoj orbiti. Ovime se dobiva efekt sličan rojenju pčela oko košnice ili kukaca oko uličnе rasvjete. Jednostavan primjer rojenja svjetlosnih čestica vidljiv je na slici 3.3.

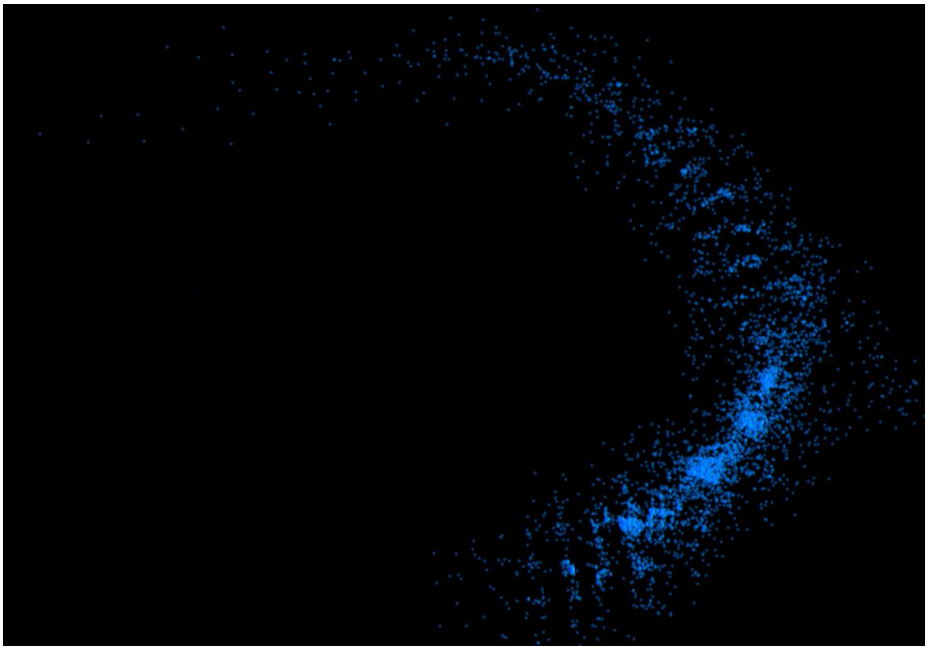
3.2. Praćenje putanje

Ovaj efekt ostvaruje jednostavnom primjenom pravila opisanog u poglavlju 2.2.5. Dobiveni efekt sličan je kretanju roja u prirodi te omogućava definiranje proizvoljne



Slika 3.1: Rojenje čestica

putanje kretanja koja u prirodi nikada ne bi mogla ostvarena. Na slici 3.2 može se vidjeti primjer roja čestica koji prati zadanu putanju.



Slika 3.2: Roj čestica prati zadanu krivulju

3.3. Objektom definirana formacija jata

U prirodi se često može vidjeti pojava gdje životinje spontano formiraju zanimljive oblike. Tako primjerice manje ribe mogu formirati jata sačinjena od mnoštva riba koja svojim izgledom podsjećaju na jednu veliku ribu. To im može pomoći da se zaštite od predatora. Takva ponašanja izazivaju veliku pažnju ljudi te se često koriste u filmovima, reklamama te video igricama pri ostvarivanju puno složenijih formacija. Primjer jednog takvog ostvarenja vidljiv je na slici 3.3. na kojoj je prikazana scena iz filma Mumija na kojoj pješčana oluja poprima oblik ljudskog lica.

Kako bi se postiglo da čestice ravnomjerno poprime strukturu nekog složenijeg objekta, potrebno je svakom boidu definirati gravitacijsku točku kojoj on treba težiti. Boid tada orbitira oko te točke zahvaljujući pravilima slijeđenja cilja i ostalim pravilima kretanja jata. Za definiranje položaja pojedinih točaka može se iskoristiti zapis poligonalne mreže nekog trodimenzionalnog objekta. Objekt je potrebno ravnomjerno popuniti tim točkama kako bi se mogao jasno razaznati oblik koji čestice formiraju. Jedna od mogućnosti je da se svakom vrhu poligona pridruži po jedna ciljna točka za svaki od boida. U tom slučaju sve se točke nalaze na površini objekta i sačuvan je njegov trodimenzionalni oblik. No problem kod ovog pristupa je što ne možemo proizvoljno definirati broj točaka koji želimo imati jer on ovisi o broju točaka definiranog objekta.

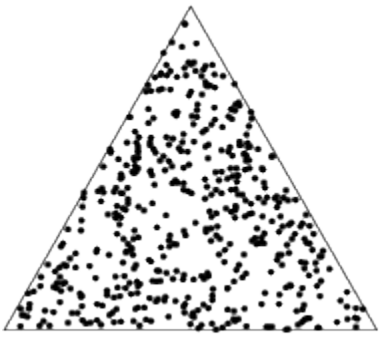
Slika 3.3: Scena iz fima Mumija

Drugi pristup bio bi uzorkovanje točaka iz poligonalne mreže objekta. Proizvoljno se može zadati pravilo uzorkovanja pojedinih poligona iz nekog skupa, te potom pravilo uzorkovanja pojedine točke iz odabranog poligona. Na ovaj način odabrane točke će i dalje ležati na površini objekta te će njegov oblik biti sačuvan. Primjerice ako je broj čestica koji je potrebno imati u sustavu deset puta manji od broja poligona objekta čiju formaciju se želi postići, poligoni objekta mogu se grupirati u skupine po deset, iz svake skupine može se nasumce odabrati jedan poligon te se iz poligona nasumce može odabrati jedna njegova točka. Na slici 3.5.a može se vidjeti primjer na kojem čestice rijetko popunjavaju strukturu objekta. Sa slike se ne može točno razaznati o kojem obliku se radi.

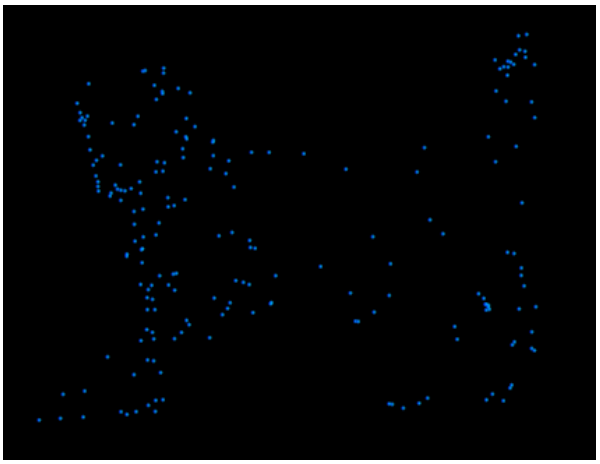
Ako je oblik formacije jednostavniji, a u sustavu se nalazi mnogo čestica, potrebno je definirati uniformno uzorkovanje čestica iz poligona. Uniformno uzorkovanje čestica iz trokuta opisano je u radu [4]. Ondje se navodi formula kojom se postiže uniformno uzorkovanje točaka unutar trokuta. Za svaki trokut definiran vrhovima A, B i C uz pomoć dva nasumično generirana realna broja r_1 i r_2 između 0 i 1, točku P unutar trokuta možemo dobiti evaluirajući izraz:

$$P = (1 - \sqrt{r_1})A + \sqrt{r_1}(1 - r_2)B + Cr_2\sqrt{r_1} \tag{3.1}$$

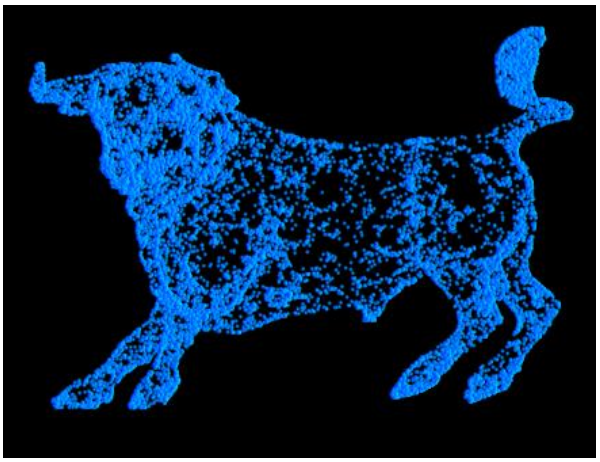
Distribucija ovako definiranih točaka prikazana je na slici 3.4. Na slici 3.5.b može se vidjeti objekt koji je gusto popunjen s česticama. Ovdje se s lakoćom može razaznati da se radi o liku bika. Nedostatak ovakvog pristupa je taj da područja s velikim brojem



Slika 3.4: Uniformno uzorkovanje točaka unutar trokuta



Slika 3.5.a: Rijetko uzorkovan oblik

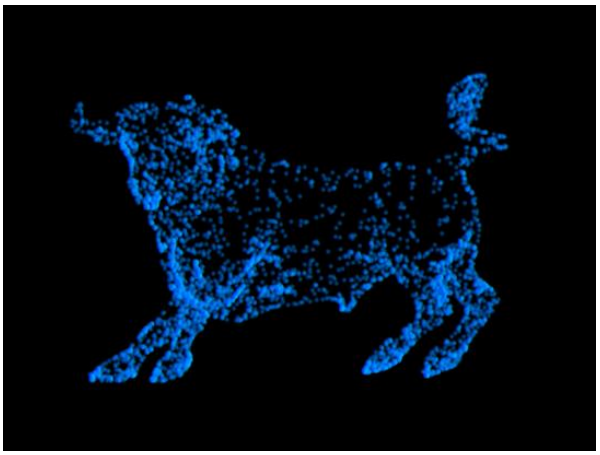


Slika 3.5.b: Gusto uzorkovan oblik

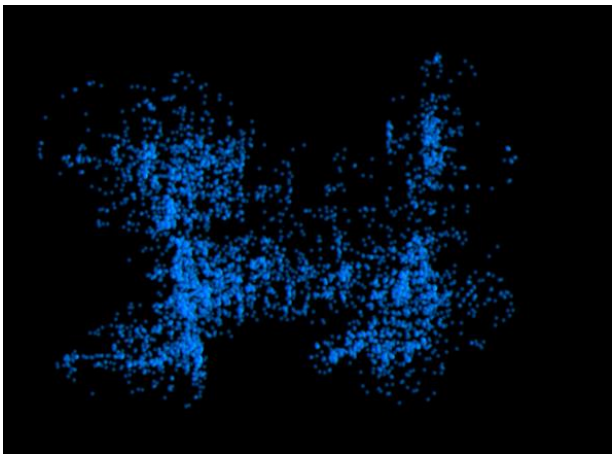
malih poligona budu gusto popunjena točkama, dok područja s malim brojem velikih poligona budu popunjena malim brojem točaka. Na slici 3.5.b. tako vidimo da glava i rep imaju veći broj točaka od trupa bika. Ovo se može izbjeći promjenom pravila za uzorkovanje poligona. Tada možemo koristiti statistički pristup pri kojem poligone odabiremo s vjerojatnošću koja je proporcionalna njihovoj površini.

3.4. Transformacije oblika

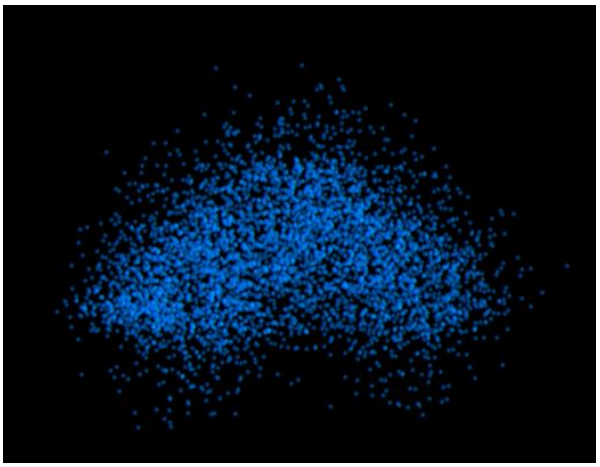
Animaciju transformacije iz jednog oblika u drugi moguće je ostvariti kombinacijom prethodne tri animacije. Ova se animacija tako može podijeliti na tri faze: početnu fazu u kojoj se odvija animacija rojenja, središnju fazu u kojoj se odvija praćenje definiranje putanje, te završne faze u kojoj se odvija faza formiranja zadanog oblika. Na samom početku čestice se postave u položaj na način da definiraju početni objekt. Kada animacija započne, čestice se počnu rojiti oko središnje točke objekta. Nakon nekog vremena može započeti druga faza u kojoj čestice slijede putanju. Nakon što svaka čestica završi slijeđenje putanje, ona se usmjerava prema završnoj točki uzorkovanoj iz drugog objekta. Ovakvom se animacijom postiže vrlo atraktivan efekt koji svojim izgledom jako podsjeća na prirodna kretanja jata i rojeva. Slike transformacije iz jednog u drugi objekt mogu se vidjeti na slikama 3.6.a, 3.6.b, 3.6.c i 3.6.d. Na prvoj slici vidi se početna formacija čestica. Na drugoj slici može se vidjeti početak rojenja, dok se na



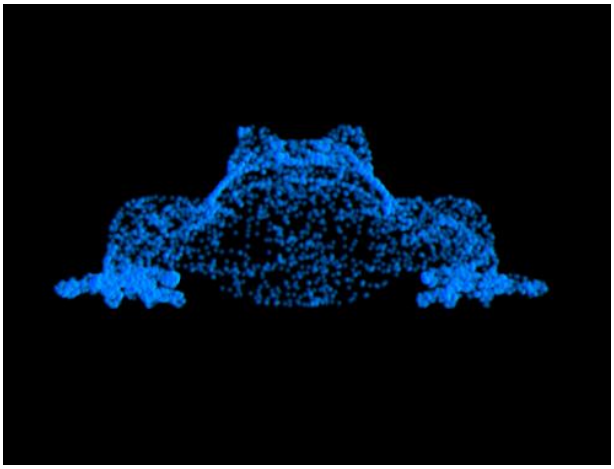
Slika 3.6.a: Početna formacija čestica



Slika 3.6.b: Početak rojenja



Slika 3.6.c: Završetak rojenj



Slika 3.6.d: Zauzimanje završne pozicije

trećoj slici vidi prijelaz iz rojenja u završnu formaciju. Formiranje drugog objekta može se vidjeti na zadnjoj slici. Animacija transformacije iz jednog oblika u drugi na sličan

način ostvarena je u radu [5]. Ondje, međutim, animacija transformacije nije podijeljena u faze, nego je ostvarena tako da je svakoj čestici pridružena jedna točka koju ta čestica mora pratiti. Također, u radu [6] je opisan pristup u kojem se postiže formiranje čestica sustava u oblik kružnice.

4. Implementacija i rezultati

Programsko rješenje implementirano je u programskom jeziku java korištenjem Java OpenGL (JOGL) biblioteke. To je biblioteka koja omogućava korištenje funkcija OpenGL (Open Grphics Library) biblioteke u programskom jeziku Java. OpenGL je višeplatformsko aplikacijsko programsko sučelje (API) koje definira niz funkcija za manipulaciju 2D i 3D vektorske grafike. Specifičnosti same implementacije opisane su u nastavku.

4.1. Učitavanje objekata i definiranje putanje

Za formiranje boida u zadani oblik potrebno je najprije odrediti način na koji će taj oblik biti predstavljen i pohranjen u računalu. Za potrebe ovog rada korišten je OBJ format zapisa objekta u svojoj pojednostavljenoj verziji. U zapisu je dovoljan popis vrhova i poligona koji definiraju poligonalnu mrežu objekta. U program je potrebno učitati dva objekta, jedan koji predstavlja početnu i jedan koji predstavlja završnu formaciju roja čestica. Program ostvaruje animaciju transformacije roja čestica iz jedne u drugu formaciju. Između te dva dijela animacije može se definirati proizvoljna putanja kojom čestice trebaju proći za vrijeme trajanja transformacije iz jednog objekta u drugi. Ta se putanja zadaje kao tekstualna datoteka koja u svakom retku sadrži x, y i z koordinate točke koja definira putanju odvojene razmacima. Za pokretanje programa potrebno je definirati sve tri ulazne datoteke, te broj čestica koje želimo u sustavu. Prilikom pokretanja program uzorkuje definirane objekte i stvara zadani broj čestica čije početne pozicije odgovaraju pozicijama uzorkovanim iz početnog objekta.

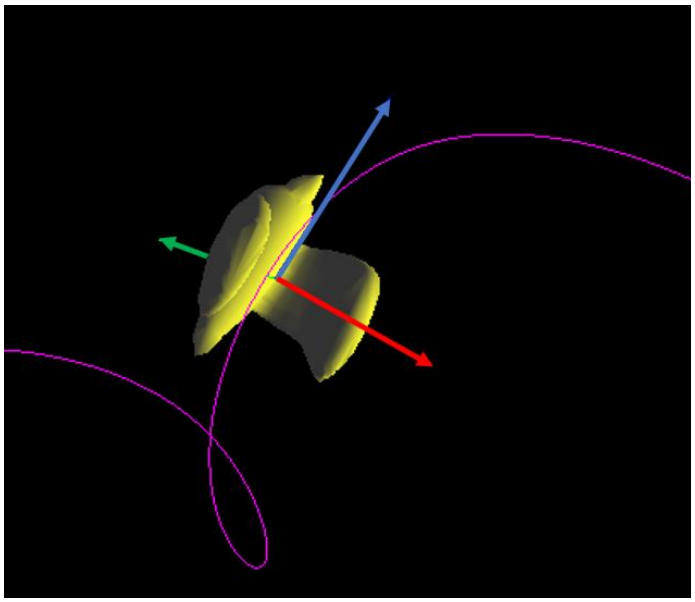
4.2. Definiranje boida

Element sustava čestica nazvan je boid. Prilikom inicijalizacije svakog boida potrebno je definirati njegovu početnu poziciju uzorkovanu iz početnog objekta, krivulju kojom se treba kretati te završnu poziciju odredišnog objekta. Svaki je boid u nekom trenutku definiran pozicijom i vektorom brzine. Na boid u svakom trenutku utječu sile definirane

pravilima kretanja boida. Rezultantni vektor sume sila koje djeluju na boid koristi se za izračunavanje brzine boida u sljedećem trenutku, dok ta brzina dalje djeluje za izračunavanje nove pozicije boida.

Svaki je boid definiran i mnoštvom parametara koji određuju njegovo ponašanje. Neki od značajnijih parametara su polumjer sfere unutar koje boid traži svoje susjede, minimalna i maksimalna brzina koju on može postići, faktori utjecaja pravila kohezije, separacije i poravnavanja i veličina samog boida.

Boid može biti prikazan na razne načine. U dvodimenzionalnom prostoru jedan od načina za prikaz boida je korištenje 2D teksture kojom se prikazuju jednostavnije čestice poput svjetlosne čestice koja u mnoštvu podsjeća na roj krijesnica. Boid je moguće prikazati i animiranom 2D teksturom. Nedostatak ovakvog pristupa je što složeniji izgled boida nije moguće ostvariti bez da cijeli sustav izgleda neprirodno. Tako primjerice ako ovim pristupom želimo prikazati roj leptira dobijemo dojam kao da odozgo gledamo na leptire koji lete vodoravno. Složenije boide u trodimenzionalnom sustavu scene moguće je prikazati nekim od složenijih 3D objekata. Boid posjeduje sve informacije potrebne za određivanje svojeg položaja i orijentacije. Na slici 4.1. vidimo primjer reprezentacije boida složenijim oblikom. Boid se orijentira prema vektoru koji određuje smjer kretanja (prikazan plavom strelicom na slici).



Slika 4.1: Orijentacija boida u 3D prostoru

4.3. Ažuriranje položaja čestice

Položaj čestice ažurira se u svakoj iteraciji glavne petlje koju izvodi program. Za potrebe izračunavanja novog položaja čestice potrebno je prvo odrediti sve sile koje na tu česticu djeluju. Sile su izražene vektorima, a predstavljaju utjecaj pojedinih pravila opisanih u poglavlju 2.2. Kako sva pravila ne moraju imati jednak utjecaj na kretanje boida, utjecaj pojedinog pravila pomnožen je faktorom pravila. Za ukupnu silu koja djeluje na boid u nekom trenutku tada dobijemo izraz:

$$\overrightarrow{F_{uk}}(t) = \sum_{i=1}^{p} k_i \, \vec{F_i}(t) \tag{4.1}$$

gdje p određuje broj pravila, k_p određuje iznos pojedinog faktora, a $\vec{F_i}$ predstavlja vektor sile koja modelira i-to pravilo. Na temelju definirane mase boida i poznate sile, možemo izračunati akceleraciju boida u određenom trenutku:

$$\vec{a}(t) = \overrightarrow{F_{uk}}(t)\frac{1}{m} \tag{4.2}$$

Iz poznate akceleracije, na temelju brzine u prethodnom trenutku, brzina u sljedećem trenutku računa se temeljem izraza:

$$\vec{v}(t) = \vec{v}(t-1) + \Delta t \cdot \vec{a}\,(t) \tag{4.3}$$

Konačno, novi položaj čestice računa se pomoću izraza:

$$\vec{x}(t) = \vec{x}(t-1) + \Delta t \cdot \vec{v}\,(t) \tag{4.4}$$

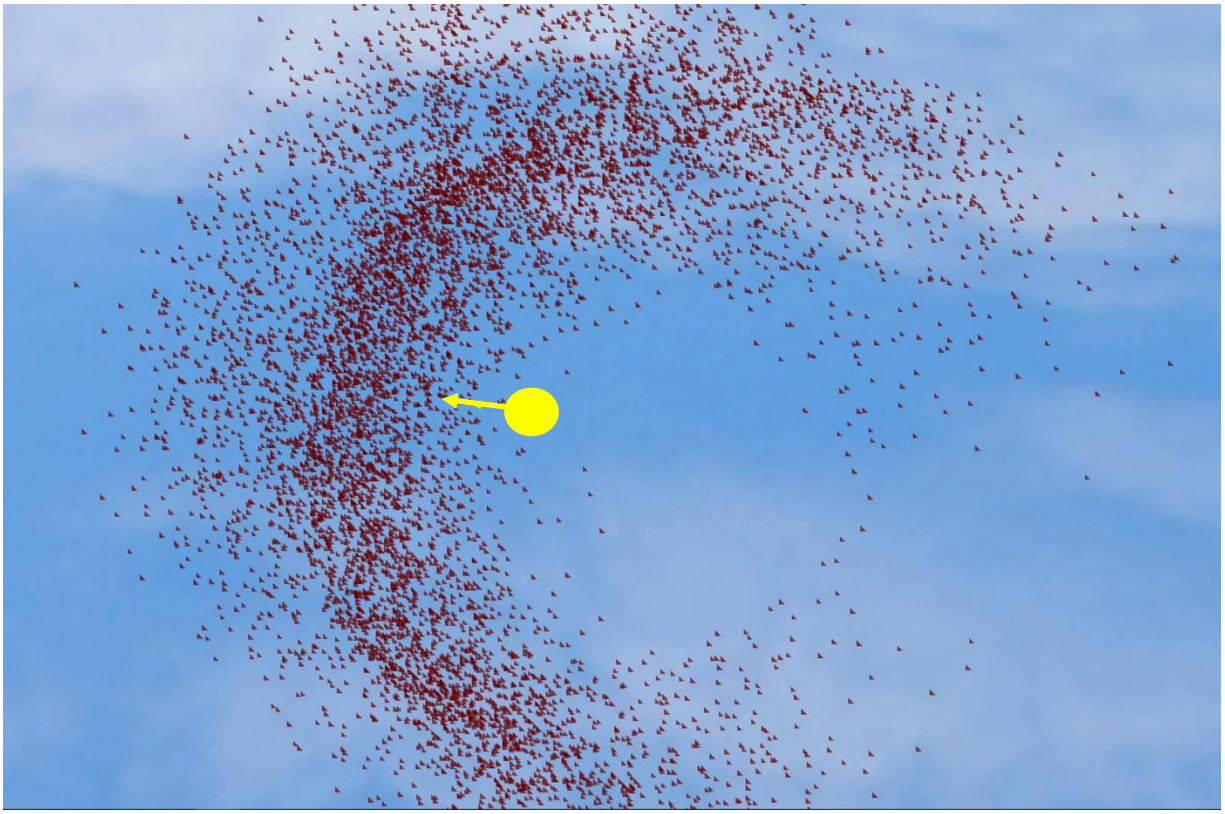
4.4. Predator

Uz jato boida koji se kreću temeljem definiranih pravila, u sustav je dodana i posebna vrsta boida: predator. Svrha uvođenja ovog elementa u sustav bilo je istraživanje podržavaju li definirana pravila nepredviđena kretanja uzrokovana vanjskom pobudom. Zbog toga, u sustav je dodan predator. Predatorom u sustavu upravlja korisnik pomoću pokazivača. Željeno ponašanje koje se željelo postići je da jato boida naglo promijeni oblik i smjer kretanja pri nailasku na predatora. Ovo se postiglo značajnim povećanjem utjecaja pravila separacije na kretanje boida koje na njega ostvaruje predator ukoliko

se nalazi u susjedstvu boida. Primjer utjecaja predatora na jato boida vidljiv je na slici 4.2. Ondje je predator predstavljen žutim krugom te strelicom koja označava njegov smjer kretanja. Sa slike je vidljivo da boidi koji se nalaze ispred predatora nastoje pobjeći ubrzavanjem i skretanjem. Zbog toga dolazi do povećane koncentracije boida ispred predatora. Nakon što opasnost prođe, boidi se polako vraćaju te ponovno počinju uspostavljati jato iza predatora.

4.5. Performanse

Pri modeliranju boida jednostavnijim grafičkim elementima, performanse sustava ponajviše ovise o veličini jata. Jedan od razloga tome je jednostavan: što je više elemenata u sustavu, to više proračuna moramo raditi prilikom ažuriranja njihovih pozicija. Ukoliko se želi imati veliki broj čestica u sustavu, potrebno je ostvariti dovoljnu brzinu izračuna kako bi brzina osvježavanja ekrana, a samim time i izgled animacije bili



Slika 4.2: Utjecaj predatora na jato boida

prihvatljivi. Zbog toga je potrebno ubrzati izračune. Izračuni pozicije boida ne mogu se previše ubrzati, ali moguće je paralelizirati izračune na više jezgara. Drugi i bitniji utjecaj na performanse sustava ima izračunavanje skupa boida koji čine susjedstvo nekom boidu. Naivni pristup ovome za svakog boida prolazi cijelim skupom boida u potrazi za onima koji zadovoljavaju uvjet susjedstva. Bolji pristup ostvaruje se korištenjem k-d stabala opisanih u 2.3.1. Ovim pristupom značajno se poboljšavaju performanse sustava što je i vidljivo iz tablice 4.1. Ondje se može vidjeti kako broj sličica u sekundi koje sustav može prikazati ovisi o broju čestica sustava i dubini k-d stabla. Vidljivo je da za sustave s manje od tisuću čestica nema potrebe za korištenjem k-d stabla, dok u sustavima s većim brojem čestica raste potreba za povećanjem dubine k-d stabla. Mjerenja su rađena na jednoj jezgri Intel i-5 procesora uz 2.6 GHz i 6GB RAM-a.

Tablica 4.1: Broj sličica u sekundi (FPS) u ovisnosti o broju čestica n i dubini k-d stabla d

d	n			
	1000	**3000**	**5000**	**8000**
1	60	15	6	2
2	60	23	10	4
3	60	38	17	8
4	60	59	27	13
5	60	60	39	20
8	60	60	60	60
10	60	60	60	60

5. Zaključak

Računalno generirani specijalni efekti na filmskom platnu u novije vrijeme privlače sve veću pozornost kako gledatelja, tako i struke. Tako danas u okviru filmskih potreba više ne postoji nemoguće. Uporabom sustava čestica na vrlo jednostavan se način mogu ostvariti atraktivni efekti. Ti sustavi mogu se definirati velikim brojem parametara te je za ostvarivanje specifičnog efekta ponekad potrebno dobro odrediti njihove vrijednosti. Njih ponekad može biti malo teže odrediti, no prednost je posjedovanje sustava koji može simulirati veliki broj efekata temeljenih na zajedničkim principima.

Vrlo je važno obratiti pozornost na algoritme korištene pri implementaciji takvih sustava. Zbog velikog broja elemenata u sustavu povećava se vrijeme osvježavanja slike što je posebno bitno u sustavima koji su napravljeni za rad u stvarnom vremenu. Zbog toga je potrebno grupirati dijelove programskog koda u nezavisne jedinice kako bi se omogućila njihova paralelizacija te pokušati izbaciti što je više moguće nepotrebnih operacija korištenjem naprednijih algoritama i struktura podataka.

6. Literatura

[1] Ackerman J., The Genius of Birds, Penguin Press, 2016.

[2] Craig W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph. 21, 4 (August 1987), stranice 25-34.

DOI: https://doi.org/10.1145/37402.37406

[3] Jae Moon Lee; Se Hong Cho; Rafael A. Calvo, A fast algorithm for simulation of flocking behavior, International IEEE Consumer Electronics Society's Games Innovations Conference, London, UK, 2009.

DOI: https://doi.org/10.1109/ICEGIC.2009.5293611

[4] Osada R.; Funkhouser T.; Chazelle B.; Dobkin D., Shape Dsitributions, ACM Transactions on Graphics (TOG), 21, 4 (October 2002), stranice 807-832.

DOI: https://doi.org/10.1145/571647.571648

[5] Xu J.; Jin X.; Yu Y.; Shen T.; Zhou M., Shape-constrained flock animation, Computer Animation and Virtual Worlds, 19, 3-4, (July 2008), stranice 319-330.

DOI: https://doi.org/10.1002/cav.231

[6] Brave A.; Nene M. J.; Autonomous formation control and target tracking in distributed Multi-agent System, International Conference on Advances in Computing, Communications and Informatics (ICACCI), New Delhi, India, 2014

DOI: https://doi.org/10.1109/ICACCI.2014.6968589

Sustavi čestica u simulacijama mnoštva i jata

Sažetak

U radu su opisani algoritmi za simuliranje mnoštva i jata objekata temeljeni na primjeni sustava čestica. Opisan je postupak postizanja željenih formacija. Pojašnjena je uloga parametara na oblikovanje ponašanja sustava. Analizirana su svojstva algoritama te je ukazano na važnost optimizacijskih metoda pri implementaciji ovakvih sustava pri izvođenju u stvarnom vremenu. Razvijeno je programsko rješenje koje demonstrira navedene algoritme.

Ključne riječi: računalna animacija, sustavi čestica, jato, roj, formacija, k-d stablo, boid

Particle Engine for Crowd and Flock Simulation

Abstract

The paper describes algorithms for simulating the flocking behaviour based on the use of particle systems. The process of achieving desired shape constrained formations is described. The role of parameters on system behavior has been clarified. Algorithm properties were analyzed and the importance of optimization methods for the implementation of such systems in real-time applications was emphasized. A software implementation that demonstrates these algorithms was described.

Key words: computer animation, particle systems, flock, swarm, formation, k-d tree, boid